REMARKS

Claims 1-25 are currently pending in the subject application and are presently under consideration. Claims 1, 9, 15, 20, and 22-23 have been amended as shown on pp. 1 and 3-5 of the Reply.

Favorable reconsideration of the subject patent application is respectfully requested in view of the comments and amendments herein.

I. 7-19-07 Telephonic Interview

Initially, applicants' representative wishes to gratefully acknowledge the Examiner's consideration of the present application *via* a telephonic interview conducted July, 19, 2007. In this regard, applicant's representative appreciates the discussion of the limitation "***source-level specified specification***" in light of Rickel *et al.* as it pertains to the claimed subject matter and as further described below.

II. Rejection of Claims 1-16, 22 and 23 Under 35 U.S.C. § 101

Claims 1-16, 22 and 23 stand rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. Claims 1, 15, and 22-23 are the independent claims. This rejection should be withdrawn for at least the following reasons. The Federal Circuit has clearly established in *Eolas Techs., Inc. v. Microsoft Corp.*, 399 F.3d 1325, 1338 (Fed. Cir. 2005) and *AT&T Corp. v. Excel Communications, Inc.*, 172 F.3d 1352, 1358. (Fed. Cir. 1999) that inventions such as that claimed by applicants are statutory.

> This court must also decide whether software code made in the United States and exported abroad is a "component of a patented invention" under 271(f)... § 271(f) refers to "components of a patented invention."... Title 35, § 101, explains that an invention includes "any new and useful process, machine, manufacture or composition of matter."... Without question, ***software code alone qualifies as an invention eligible for patenting under these categories***, at least as processes. *Eolas Techs., Inc. v. Microsoft Corp.*, 399 F.3d 1325, 1338 (Fed. Cir. 2005). (emphasis added).

The Examiner contends that the claimed subject matter is non-statutory because it amounts to software or descriptive material per se, which does not fall within any category of statutory subject matter. While Examiner agrees that software is patentable, either alone as a method or process (e.g., withdrawal of the § 101 rejection of claims 17 and 20 and associated dependent claims), or as attached to a computer-readable medium (e.g., withdrawal of the § 101 rejection of claim 24), Examiner contends that the subject claims do not recite a computer-readable medium or other hardware components that would permit the functionality of the claimed subject matter to be realized.

Applicants' representative respectfully disagrees with the Examiner's contentions and submits that the Examiner is misconstruing the requirements necessary to fulfill the conditions for patentability under 35 U.S.C. § 101. The Federal Circuit in *Eolas Techs., Inc. v. Microsoft Corp.* clearly established that software code alone is statutory subject matter, ***at least*** as processes. However, the allowance for software code alone as processes in *Eolas* does not limit the otherwise patentable subject matter categories.

For example, Claims 1-16, and 22-23 claim executable code check systems or manufactures. Systems, processes, and manufactures are by themselves statutory subject matter. By the standards set forth in the above decision, a computer implemented system or process, in the form of software, hardware, or the combination of both clearly falls within the categories of statutory subject matter. Furthermore, the subject claims produce a useful, concrete, and tangible result.

> Because the claimed process [method] applies the Boolean principle [abstract idea] ***to produce a useful, concrete, tangible result*** ... on its face the claimed process comfortably falls within the scope of § 101. *AT&T Corp. v. Excel Communications, Inc.*, 172 F.3d 1352, 1358. (Fed. Cir. 1999) (Emphasis added); *See State Street Bank & Trust Co. v. Signature Fin. Group, Inc.*, 149 F.3d 1368, 1373, 47 USPQ2d 1596, 1601 (Fed.Cir.1998). The inquiry into patentability requires an examination of the contested claims to see if the claimed subject matter, as a whole, is a disembodied mathematical concept representing nothing more than a "law of nature" or an "abstract idea," or if the mathematical concept has been ***reduced to some practical application rendering it "useful."*** *AT&T* at 1357 *citing In re Alappat*, 33 F.3d 1526, 31 1544, 31 U.S.P.Q.2D (BNA) 1545, 1557 (Fed. Cir. 1994) (emphasis added).

As provided above, the legal standard set forth by the Federal Circuit in *A&T Corp. v. Excel Communications, Inc.* for determining whether a claim is directed to statutory subject matter is whether a claim can be applied in a practical application to produce a useful, concrete, and tangible result. It is the result of the claims as applied in a practical application that is germane to the determination of whether the claims are directed towards statutory subject matter, not whether the underlying claims contain physical limitations (e.g., the computer readable medium embodiment or hardware components that would permit the functionality of the claimed subject matter). The subject claims clearly satisfy this legal standard.

For example, regarding independent claims 1 and 15, the claims relate to systems that determine whether a fault condition exists in executable code and provide information if a fault condition is determined to exist. Providing information based on determination of a fault condition is a useful, concrete and tangible result. While applicants' representative contends that such systems are inherently directed to computer-related entities (e.g., by receiving and checking executable code object files), for the avoidance of doubt, claims 1 and 15 have been amended to indicate that the claims relate to executable code check ***computing*** systems that provide information if a fault condition is determined. The claimed invention provides exposure of executable object code faults according to the provided framework, which can then be corrected. As described above, these claims recite an invention that produces a useful, concrete, and tangible result which clearly satisfies the standard set forth by the Federal Circuit for statutory subject matter under 35 U.S.C. § 101.

Regarding independent claims 22 and 23, these claims similarly recite a ***specification providing information to be employed to statically check the executable code***. However, Examiner contends that the data packet as recited amounts to software or descriptive material per se, which does not fall within any category of statutory subject matter, and that the claims fail to recite a computer-readable medium or other hardware components that would permit the functionality of the claimed subject matter to be realized. Applicants' specification p.5, ll. 24-26 is cited for support that recited "computer components" are described as software. Applicants' representative respectfully disagrees with Examiner's contentions and submits that the term "computer

component" is intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. *Id.* Furthermore, claims 22 and 23 recite that the data packet is transmitted between two or more computer components, which can be hardware or a hardware-software combination. While applicant's representative contends that it is inherent to the accomplishment of transmission of the packets, for the avoidance of doubt, claims 22 and 23 have been amended to recite data packets transmitted between two or more computer components ***over a computer-readable transmission medium***. Providing such information to facilitate static checking of executable code produces a useful, concrete, and tangible result which clearly satisfies the standard set forth by the Federal Circuit for statutory subject matter under 35 U.S.C. § 101.

In view of at least the foregoing, it is readily apparent that applicants' invention as recited in independent claims 1, 15, 22, and 23 (and associated dependent claims 2-14, and 16) recite statutory subject matter and produce a useful, concrete, and tangible result. Accordingly, withdrawal of this rejection is respectfully requested in view of the foregoing comments.

III. Rejection of Claims 1-8 and 11-25 Under 35 U.S.C. §103(a)

Claims 1-8 and 11-25 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over disclosed art of record DeLine *et al.*, "Enforcing High-Level Protocols in Low-Level Software," in view of Rickel *et al.*, U.S. 5,854,924. Claims 1, 15, 17, 20, and 22-25 are the independent claims. Without conceding the propriety of the combination, applicants' representative respectfully request withdrawal of this rejection, because DeLine *et al.*, alone or in combination with Rickel *et al.*, does not teach or suggest each and every limitation of applicants' claimed invention.

> To reject claims in an application under § 103, an examiner must establish a *prima facie* case of obviousness. A *prima facie* case of obviousness is established by a showing of three basic criteria. First, there must be some apparent reason to combine the known elements in the fashion claimed by the patent at issue (e.g., in the references themselves, interrelated teachings of multiple patents, the effects of demands known to the design community or

> present in the marketplace, or in the knowledge generally available to one of ordinary skill in the art. To facilitate review, this analysis should be made explicit. Second, there must be a reasonable expectation of success. ***Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations.*** *See* MPEP § 706.02(j). *See also KSR Int'l Co. v. Teleflex, Inc.*, 550 U. S. ____, 04-1350, slip op. at 14 (2007). The reasonable expectation of success must be found in the prior art and not based on applicant's disclosure. *See In re Vaeck,* 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991) (emphasis added).

The present invention relates to static checking of object code, and more particularly, to a system and method employing pre- and/or post- condition(s) specified at a source code level and persisted (e.g., in associated object code and/or a specification repository) facilitating static checking of the object code. The system and method are based, at least in part, upon a framework that employs rules for using an interface to be recorded as declarative specifications in an existing language. The executable code (e.g., object code) with specification(s) is then provided to a checker. The checker employs the specification to facilitate static checking of the object file and provides information if a fault condition (e.g., error(s)) is determined to exist. As a result, programming error(s) that could cause run-time exception(s) and that could escape traditional testing methods (e.g., static source code checking at compile time) can be mitigated.

In contrast, the Vault programming language disclosed in DeLine *et al.* allows a programmer to describe resource management protocols that the compiler can statically enforce through the use of an abstract global state of the program as it is compiled and type guards used by a type checker at compile-time. (*See* page 1, Abstract and § 1, "Vault programming language provides . . . " *et seq.*). "The global state, called the held-key set, consists of a set of keys, which are simply compile-time tokens representing run-time resources." (*See* page 2, § 2.1, ¶ 1). Type guards can be used by a programmer to specify domain-specific resource management protocols. (*See* page 1, § 1, "Vault programming language provides . . . " *et seq.*). In a further passage, DeLine *et al.* describes the usage of the Vault programming language as writing an interface specification in Vault, translating an existing driver into Vault code from C to perform the compile-time type checking, then recompiling the code back to C as an end product. (*See* page 8, § 4, ¶ 7).

As a result, Deline *et al.* does not receive an object file.

Although Examiner concedes that DeLine *et al.* does not expressly disclose that the received file "having an embedded specification is an object file (Official Action dated February 8, 2007, p. 10), applicants' representative submits that DeLine *et al.* teaches away from a persisted embedded specification in the object file as claimed. Notably, "[k]eys are ***purely compile-time entities*** that ***have no impact on runtime representations or execution time***." (*See* page 2, § 2.1, ¶ 1) (emphasis added). Furthermore, "[t]ype guards ***have no impact on run-time representation*** or execution time." (*See* page 2, § 2.1, ¶ 2) (emphasis added).

In response, Examiner disagrees and contends that DeLine *et al.* "teaches a file that includes a persisted embedded specification (*see*, for example, DeLine *et al.*, p. 1, § 1, ¶ 2, "The Vault programming language . . . ." *et seq.*)", but concedes that DeLine *et al.* does not disclose that the file is an object file. (Final Office Action dated June 21, 2007, p. 3, "As set forth in the Office action, . . ." *et seq.*). Thus, it is contended that the elements of the specification being "purely compile-time entities that have no impact on runtime representations" is not evidence that embedding the specification in the object file would render the system unsatisfactory for its intended purpose, preclude embedding the specification in the object file, or change the DeLine *et al.* principle of operation. *Id.* In a subsequent rejection of claim 2, it is contended that "purely compile-time entities . . ." "shows that the embedded specification is removed from the object file." (Final Office Action dated June 21, 2007, p. 8, "With respect to claim 2 . . . ." *et seq.*).

Applicants' representative respectfully disagrees and contends that, although Deline *et al.* may not preclude embedding the specification in the object file, it would render the system inoperable for its intended purpose where an intended purpose can be fairly read to ensure no impact on runtime representations or execution time. Specifically, under a fair reading, DeLine *et al.* suggests that there is a benefit to removing the source code annotations (e.g., by using such language as "***purely*** compile-time entities" and "there is ***no*** impact on run-time representation or execution time."). (*See* page 2, § 2.1, ¶ 1-2) (emphasis added). Accordingly, applicants' representative respectfully submits that Deline *et al.* specifically employs ***purely compile-time entities*** to avoid impacts on runtime representations or execution time caused by lingering annotations. Moreover, persisting specifications in the resulting object file, would either change the principle of

operation of DeLine *et al.* as a compile-time type checking system which removes any annotations from the runtime representations, or would at least be extraneous or irrelevant to the operation, because Deline *et al.* merely describes a compile-time checking system. (*See* p. 2, § 2.1, ¶ 1-2 and page 8, § 4, ¶ 7).

Consequently, applicants' representative contends that the salient design features and principles of operation of the DeLine *et al.* system is that it operates at compile-time on a source file that has annotations such as type guards, which are used by a type checker, and which are absent from the resulting object file. Additionally, applicants' representative contends that while DeLine *et al.* may contain a type of "specification" in the source file, it is not persisted in the sense that a source-level specification is persisted in the object file or repository of applicants' claimed invention. As a result, any type of DeLine *et al.* "specification" in the source file fails to teach or suggest the applicants claimed embedded specification in the object file.

> If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims prima facie obvious. *In re Ratti*, 270 F.2d 810, 123 USPQ 349 (CCPA 1959). *See* MPEP § 2143.02 [R-5] VI. Patentee taught the device required rigidity for operation, whereas the claimed invention required resiliency. The court reversed the rejection holding the "suggested combination of references would require a substantial reconstruction and redesign of the elements shown in [the primary reference] as well as a change in the basic principle under which the [primary reference] construction was designed to operate." *Id.*

Accordingly, applicants' representative contends that suggested modifications to DeLine *et al.* are too substantial as to render the applicants' claimed invention prima facie obvious. For example, such modifications as persisting specifications to an object file (and in some case embedded in the object file) and static checking of the object file would require a substantial redesign of the elements of the DeLine *et al.* system in order to render applicants' claimed invention obvious. Moreover, such modifications would change the basic principle of operation of DeLine *et al.* from that of a compile-time

source code check that removes the purely compile time entities to a post compile-time object file check.

Rickel *et al.* cannot be said to cure this deficiency. Rickel *et al.* merely discloses a debugging tool for statically debugging a representation of a binary program. *See* Abstract. Such object file representations include a file created by a decompiler, or a symbolic or other representation of the object file. (*See* col. 3, ll. 53-68 and col. 2, ll. 29-36). This representation is then analyzed for errors by the Rickel *et al.* system. While Examiner does not explicitly disagree with the applicants' representative's contention that "'a representation of a binary file' is not the object file according to applicants' invention," it is contended that the disclosure of intermediate files or representations of the binary file is sufficient to preclude patentability of applicants' claimed invention. (Final Office Action dated June 21, 2007, p. 4, "The binary program . . . ." et seq.). Applicants' representative construes this as an assertion that the applicants' claimed static checking of the object file (with persisted specification either embedded or stored) is an obvious modification to the Rickel *et al.* (either alone or in combination with DeLine *et al.*) static analysis of the object file representation.

Applicants' representative disagrees and contends that such "representations" are essential to Rickel *et al.* as a precursor to the remaining debugging analysis. *Id.* For example, the Rickel *et al.* back end analyzer is specifically configured to analyze decompiled or otherwise intermediate representations of object files. "The power of the described static debugging tool is primarily in the back end analyzer . . . . [which] is arranged to receive and analyze the machine-independent intermediate file." (*See* col. 4, ll. 26-33). Forcing the Rickel *et al.* back-end analyzer to operate on a non-machine-independent object file, would change the operation of the Rickel *et al.* system, and would render the system inoperable for its intended purpose (e.g., "to analyze a wide variety of binary program files regardless of the particular machine for which the binary program file being debugged was created."). *Id.*

Moreover, Rickel *et al.* is silent with respect to performing a debugging analysis of the received object file except through the analysis of such intermediate or representative forms created by a decompiler or otherwise. Furthermore, Rickel *et al.* is silent with respect to any persisted source-level specified specification information either embedded in the object file or in a repository according applicants' invention. Although

column 3, lines 18 – 26 and column 4, lines 35 – 39 are cited for support that Rickel *et al.* discloses receiving an object file in executable code check system, employing a specification to facilitate static checking of the object file, and providing information if a fault condition is determined, applicants' representative submits that the Rickel *et al.* analysis of a representation of the binary file does not teach or suggest a static checking of the object file according to applicants' claimed invention. Furthermore, the Rickel *et al.* system call and restrictions library file used to provide system or machine specific information to the intermediate file analyzer does not teach or suggest source-level specified specification persisted to a storage repository or embedded in the object file as claimed in applicants invention. (*See* column 4, lines 13 – 28).

Consequently, applicants' representative contends that the salient design features and principles of operation of the Rickel *et al.* system is that it operates on a decompiled, symbolic representation, or otherwise representative version of a binary program file to analyze errors or potential errors in the representation of the binary program file. The Rickel *et al.* system may alternatively use a hint file to assist in decompilation or a system call and restrictions library file for providing information specific to each of a variety of particular systems or machines with which the binary program file is designed to be used.

Accordingly, applicants' representative submits that suggested modifications to Rickel *et al.* are too substantial as to render the applicants' claimed invention prima facie obvious. For example, modifications would be required to redesign the Rickel *et al.* analyzer element from one that is configured to analyze the binary program file representation (e.g., a decompiled object file or machine independent intermediate file) to one that analyzes the object file itself. Additionally, the resulting redesign would change the principle of operation such that the decompilation operation and hint file are obviated, and would render Rickel *et al.* inoperable for its intended purpose.

In sum, the combination of DeLine *et al.* with Rickel *et al.* fails to teach all the limitations of applications invention as claimed. Moreover, the resulting combination fails to suggest the applicants' claimed invention – the resulting combination, without more, would produce a debugger of representations of object code (Rickel *et al.*) that does not have persisted specifications in the object code (DeLine et al.). Furthermore, any suggested modification required to teach or suggest applicants' claimed invention would

require redesign of salient features of the art references' elements that are too substantial, either individually or in combination, such that the teachings of the references are not sufficient to render the claims prima facie obvious. Alternatively, such proposed modifications would render the references inoperable for their respective intended purposes.

Specifically, regarding independent claim 1 (as well independent claims 17, 22, 24, and 25 that recite similar features) recites ***receives an object file having an embedded specification***. Because DeLine *et al.* neither receives an object file nor has embedded specifications in the object file, as described above, DeLine *et al.* cannot be said to teach or suggest this limitation. Furthermore, because Rickel *et al.* is silent with respect to embedded specifications, Rickel *et al.*, either alone or in combination with DeLine *et al.* cannot be said to teach or suggest this limitation. Moreover, while Examiner contends that the motivation to modify the system of DeLine *et al.* to operate on an object file (as it is contended that Rickel et al. suggests), is provided by the desire to modify DeLine *et al.* such that it is independent of the original programming language. However, because DeLine *et al.* is a compile-time source file type checker, such modifications are non-trivial to the point where the principle of operation is so substantially altered that the teachings of the reference are not sufficient to render the claims prima facie obvious. Furthermore one skilled in the art would not be motivated to make DeLine *et al.* independent of the original programming language, because the essence of DeLine *et al.* is that it operates at compile-time on the original programming language source file. As a result, DeLine *et al.* is inextricably linked to the original programming language.

Regarding independent claims 15, 20, and 23, these claims similarly recite ***static checking of the object file or executable code*** and a ***source-level specified specification***. Because DeLine *et al.* does not receive or check an object file, as described above, DeLine *et al.* cannot be said to teach or suggest this limitation. Although DeLine *et al.* specifies type guards used at the source level, the type guards are purely compile-time entities that do not persist after compilation. Furthermore, because Rickel *et al.* checks only an intermediate file representation or a symbolic representation of an object file, Rickel *et al.*, either alone or in combination with DeLine *et al.* cannot be said to teach or suggest ***static checking of the object file or executable code***. Moreover, the Rickel *et al.* system call and restrictions library file used to provide system or machine specific

information to the intermediate file analyzer does not teach or suggest ***source-level specified specification*** persisted to a storage repository or embedded in the object file as claimed in applicants invention. Additionally, while Examiner contends that the motivation to provide DeLine *et al.* with a Rickel *et al.* repository exists from a desire to flexibly retrieve the specification from a repository external to the object file, the proposed motivation fails to recognize that DeLine *et al.* does not operate on object files as described above. Furthermore, the proposed motivation fails to recognize that Rickel *et al.* does not operate on object files, rather it operates on intermediate representation of object files, with any specifications (such as the system call and restrictions library file) not specified at source level and not persisted after the received object file was compiled. Accordingly, the suggested modifications to Rickel *et al.* are too substantial as to render the applicants' claimed invention prima facie obvious. The modifications would change the principle of operation by requiring redesign of the Rickel *et al.* analyzer element from one that is configured to analyze the intermediate object file representation (e.g., a decompiled object file or machine independent intermediate file) to one that analyzes the object file itself. Additionally, such changes would render Rickel *et al.* inoperable for its intended purpose.

Reconsideration and withdrawal of the rejection of claim 1, 15, 17, 20, and 22-25 (and associated dependent claims 2-14, 16, 18-19, 21, and 23) under 35 U.S.C. § 103(a) is respectfully requested in view of the comments above.

IV. Rejection of Claims 9 and 10 Under 35 U.S.C. § 103(a)

Claims 9 and 10 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over DeLine *et al.* in view of Rickel *et al.*, as applied to claim 1, and further in view of Alaluf, U.S. 2004/0230958. Claims 9 and 10 directly depend from claim 1. Without conceding the propriety of the of DeLine *et al.* in view of Rickel *et al.*, and further in view of Alaluf, for at least the reasons set forth for claim 1 above, reconsideration and withdrawal of the rejection is respectfully requested.

CONCLUSION

The present application is believed to be in condition for allowance in view of the above comments and amendments. A prompt action to such end is earnestly solicited.

In the event any fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP464US].

Should the Examiner believe a telephone interview would be helpful to expedite favorable prosecution, the Examiner is invited to contact applicants' undersigned representative at the telephone number below.

Respectfully submitted,

AMIN, TUROCY & CALVIN, LLP

/Himanshu S. Amin/
Himanshu S. Amin
Reg. No. 40,894

AMIN, TUROCY & CALVIN, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731